VŠB - TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2013 Josef Šeliga

VŠB - TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
KATEDRA INFORMATIKY

Osadníci z Katanu

The Settlers of Catan

2013 Josef Šeliga

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Josef Šeliga**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Osadníci z Katanu
The Settlers of Catan

Zásady pro vypracování:

Cílem práce je vytvořit systém pro hraní hry Osadníci z Katanu a jejich rozšíření v prostředí internetu. Systému se bude skládat ze serverové části, která bude umožňovat připojování jednotlivých uživatelů, sledování statistik a pořádání turnajů. Klientská část pak bude samostatná aplikace která se připojí k serveru a umožní uživateli hrát hru s ostatními hráči.

Systém umožní:
1. Správu uživatelů.
2. Sledování statistik uživatelů.
3. Pořádání turnajů.
4. Hraní hry více hráčů v prostředí internetu.
5. Využití hráče s umělou inteligencí.
6. Volby jednotlivých rozšíření hry Osadníci z Katanu (např. Námořníci, Města a Rytíři, Kupci a barbaři, ...) nebo jejich kombinace.
7. Hraní historických scénářů (varianty rozšíření).

Práce bude obsahovat:
1. Popis použitých technologií.
1. Popis navržené architektury.
2. Implementaci hry Osadníci z Katanu.
3. Programátorskou dokumentaci řešení s využitím diagramů jazyka UML.

Seznam doporučené odborné literatury:

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
[2] Osadníci z Katanu. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-10-05]. Dostupné z: http://cs.wikipedia.org/wiki/Osadn%C3%ADci_z_Katanu

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Podpis: .....

Poděkování

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Cílem práce je převést deskovou hru Osadníci z Katanu do podoby hratelné aplikace s možností hraní více hráčů v prostředí internetu a možností výběru mimo základních pravidel také rozšíření hry. V první části práce rozebírám pravidla pro jednotlivá rozšíření a jednotlivé aspekty z obecného hlediska, jenž je třeba brát v potaz při počátku vývoje. V druhé části je rozebrána konkrétní implementace této hry, popis komunikace, hrací plochy, implementace umělé inteligence do hry a další herní mechanismy v rámci zadané problematiky.

Klíčová slova: Osadníci z Katanu: Námořníci, Osadníci z Katanu, PictureBox, Graphics, umělá inteligence, pravidla hry, vlákna

Abstract

The aim of my thesis is to convert the board game „Osadníci z Katanu“ to playable application on internet. This change includes multiplayer and new game settings out of scope the standard rules. The first part of my thesis is focused on the out of scope settings and basic conditions which have to be counted at the beginning of the development. The implementation, communication description, game board, artificial intelligence implementation into the game and the other game mechanism – these aspects mentioned above are described in the second part of my thesis.

Key words: The Settlers of Catan: Sailors, The Settlers of Catan, PictureBox, Graphics, artificial Intelligence, game rules, threads

Obsah

1 Úvod

Osadníci z Katanu, je populární desková hra, která je specifická oproti jiným hrám především svým tvarem hrací plochy a celkovým herním tématem. Na hrací ploše, kterou symbolizuje ostrov, je možné pokládat cesty a stavět vesnice nebo města. Je zde kladen velký důraz na taktiku hry a předvídání akcí protihráčů. Hráč, který v průběhu hry dosáhne největšího rozvoje v osidlování ostrova, vyhrává.

Hra je celosvětově rozšířena a přeložena do několika jazyků. V rámci expanze hry, jsou také vytvářena rozšíření a vylepšení, jenž mají za úkol poskytnout hráči kvalitnější herní zážitek.

Rozhodl jsem se tuto hru převést do formy aplikace z důvodu unikátní hratelnosti a dynamičnosti oproti jiným deskovým hrám. Díky těmto vlastnostem získává hra potenciál a unikátnost v portfoliu deskových her.

Práce je rozdělena do pěti kapitol, které jsou následně podrobněji rozpracovány v podkapitolách. Mezi hlavní kapitoly patří obecná definice pravidel a hraní hry.V práci jsou rozebrána pravidla hry a rozšíření pro základní pochopení metodiky hry, další kapitola rozebírá požadavky, které je zapotřebí navrhnout před samotným počátkem vývoje. Předmětem následující části je rozbor obecné metodiky řešení zadané problematiky, kde je nastíněna architektura, způsob komunikace mezi klientskou a serverovou částí a rozbor jednotlivých funkčních aspektů. Poté následuje část implementace serverové a klientské části, v níž je již rozebrán detail vyvíjené metodiky pro každou část zvlášť. V serverové části je mimo základní charakteristiky hry (herní plocha, obchodování atd.) také podrobněji rozepsána metodika řešení umělé inteligence ve hře a její zapracování do procesu. Klientská část popisuje realizaci grafického zobrazení (herní plocha, způsob aktualizace dat, směnný obchod) a také detekce kliknutí na hrací pole hry.

Pro řešení požadavků jsem použil informace získané hraním této hry a studováním podrobných pravidel. V dnešní době digitalizace, díky převedení do formy aplikace, získáme širší okruh hráčské populace než je dosaženo pouhou fyzickou distribucí hry.

2 Pravidla a jejich rozšíření

V rámci této hry můžeme rozlišovat několik rozšíření, která upravují pravidla, případně rozšiřují stávající mechanismy. Ovšem všechna tato rozšíření jsou postavena nad základními pravidly a pouze je doplňují o jednotlivé funkční prvky, případně upravují stávající funkční mechanismy.

V rámci této práce jsou implementována pravidla:

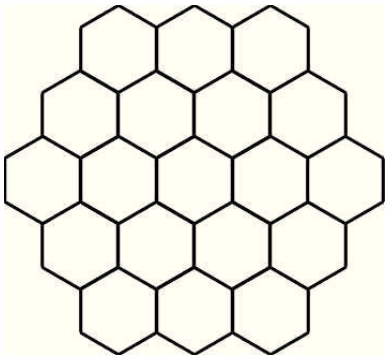
- Základní pravidla
- Rozšíření Námořníci

2.1 Hrací plocha a základní mechanismy

Hrací plocha je v této deskové hře realizována pomocí šestiúhelníků, kdy lze na hranách stavět cesty a na jednotlivých bodech lze stavět budovy. Jedno hrací pole obsahuje místo pro 6 budov a 6 cest. Každé hrací pole obsahuje materiál a číslo, které reprezentuje hodnotu při hodu kostky. V rámci jednotlivých rozšíření se vždy pracuje se základními 5 materiály + doplňkovými materiály dle rozšíření:

- Les (generuje surovinu klády)
- Pastvina (generuje surovinu ovce)
- Pole (generuje surovinu obilí)
- Pahorkatina (generuje surovinu cihly)
- Hory (generuje surovinu železo)
- Poušť (negeneruje žádnou surovinu)
- Moře (rozšíření Námořníci, negeneruje žádnou surovinu)
- Zlatá řeka (rozšíření Námořníci, generuje surovinu, která je ihned proměněna na libovolnou surovinu dle výběru)

Jako další hrací prvek nalezneme na některých bodech hrací plochy také přístavy, které nám umožňují uskutečnit směnný obchod.



Obrázek 1: Tvar hrací plochy

2.1.1 Směnárna

Přístavy nám umožňují provést obchod za výhodnějších podmínek, než základní měnící poměr 4:1 (4 totožné suroviny za 1 libovolnou). Přístavy se dále dělí na speciální a standardní. Speciální přístav je specifický tím, že je vázán na konkrétní surovinu a umožňuje použít kurz 2:1 (2 specifické suroviny za 1 libovolnou surovinu). Standardní přístav nabízí kurz 3:1 (3 totožné suroviny za 1 libovolnou), ovšem tento poměr již není limitován na konkrétní surovinu.

Dalším druhem směnného obchodu je výměna s hráči. Toto privilegium má pouze hráč na tahu, který může s ostatními hráči libovolně měnit suroviny.

2.1.2 Průběh hry

V případě počátku hry (jakmile jsou vesnice a cesty zdarma umístěny) začíná standardní průběh hry. Dle hodnoty hozené součtem dvou kostek určíme číslo pole, jehož materiál si hráči, kteří mají postaveny budovy na tomto poli, vezmou. Dle typu budovy si hráči vezmou 1 (vesnice) nebo 2 (město) suroviny daného typu. V případě hození hodnoty 7 se provádí přesunutí figurky Zloděje, popř. Piráta. Tyto mechanismy jsou rozebrány podrobněji v rozboru jednotlivých pravidel. Z nasbíraných surovin může hráč stavět objekty dle daného rozšíření(vesnice, města, cesty atd.). Jakmile hráč již nechce dále pokračovat v tahu, tah se přenese na dalšího hráče, jenž započne cyklus hodem kostkou. Jednotlivé kroky hráče na tahu můžeme obecně shrnout:

- Hod kostkami a sběr surovin.
- V případě hodu čísla 7 přesun Zloděje/Piráta.
- Stavba objektů (vesnice, město, cesta, loď), nákup speciálních karet.
- Výměna surovin (přístavy, hráči).
- Použití speciálních karet.

Ovšem zůstávají zde i akce, které provádí i hráči, jež na tahu nejsou:

- Odevzdávání karet - v případě že padne číslo 7, všichni hráči, jež mají počet karet větší než 7, musí polovinu odevzdat zpět.
- Obchodování - hráč na tahu může obchodovat i s hráči, kteří na tahu nejsou.

Hru vyhraje ten, kdo jako první dosáhne stanoveného počtu výherních bodů. Výherní body jsou počítány na základě budov, speciálních a bonusových karet. Rozložení je následovné:

- Vesnice = 1 bod
- Město = 2 body
- Bonusová karta = 2 body (dle typu rozšíření)
- Speciální karta = 1 bod (dle typu rozšíření)

2.2 Základní pravidla

Základní pravidla definují chování a funkci jednotlivých aspektů od založení až po ukončení hry. Tato pravidla také obsahují figurku Zloděj, která plní úlohu blokování suroviny. V případě vložení Zloděje na některé hrací pole a následném padnutí čísla daného pole na kostkách si nikdo nevezme suroviny nacházející se na tomto poli. Hráč, kterému padne číslo 7, má možnost přesunout figurku Zloděje a umístit ho dle svého uvážení na hrací plochu, přičemž v místě, kam ho vloží, si od jednoho z hráčů, který má na tomto poli umístěnu budovu, vezme náhodnou surovinu.

2.2.1 Typy budov a jejich stavba

V základních pravidle je možnost budovat 3 typy objektů: vesnice, města a cesty. Města a vesnice jsou důležité pro sběr surovin a postup ve hře. Ovšem abychom mohli postavit tyto budovy, je nutno mít postaveny cesty. Na počátku má každý hráč dva volné tahy, kdy v každém tahu může postavit zdarma jednu cestu a vesnici. Jediným omezením je, aby v okolí nebylo žádné město či vesnice přičemž cesta musí vždy vést od jedné z budov. V případě stavby objektů po vyčerpání volných cest a vesnic platí tato pravidla:

- Mezi jednotlivými městy, vesnicemi musí být vždy dvě cesty.
- Abychom mohli postavit město je potřeba mít postavenu vesnici, kterou městem nahradíme.
- Cestu lze stavět pouze jako navazující prvek na jinou cestu, případně na budovu.

2.2.2 Speciální karty

V základní verzi a rozšíření Námořníci, je možno získat speciální karty. Tyto karty dělíme do 3 skupin:

- Rytíř - možnost přesunutí Zloděje/Piráta ovšem již bez získání suroviny od hráče (v případě Zloděje).
- Pokrok - uděluje zvláštní privilegia při použití (stavba dvou cest zdarma, výběr dvou libovolných surovin zdarma).
- Výherní bod - přičtení 1 bodu k celkovému počtu.

2.2.3 Bonusové karty

Za každou bonusovou kartu získáme dva body k celkovému součtu bodů. Rozlišujeme dva druhy bonusových karet - dle typu rozšíření se některé bonusové karty nepoužívají:

- Nejdelší obchodní cesta - získá hráč, jenž má postavenou nejdelší nepřerušenou cestu.
- Největší vojsko - získá hráč s největším počtem použitých speciálních karet Rytíř.

2.3 Rozšíření Námořníci

Toto rozšíření neupravuje herní mechanismy, ale ke stávajícím přidává nové. Největší rozdíl je v hrací ploše, která je větší a rozšířena o dva nové materiály viz. sekce **Hrací plocha a základní mechanismy**. Moře jako takové negeneruje žádnou surovinu, ovšem v rámci hry nám dává příležitost stavět lodě, díky kterým můžeme propojovat jednotlivé ostrovy a díky tomu expandovat. V případě získání suroviny Zlatá řeka, je nutné si vzít za každou kartu, jednu libovolnou surovinu. Námořníci dále rozšiřují hru o figurku Pirát, která má obdobnou funkci jako Zloděj, ovšem je ho možno umisťovat pouze na materiálu Moře, kde blokuje přemisťování, stavění lodí. V případě umisťování Piráta, oproti Zloději, již hráč nebere žádnou kartu ostatním. Toto rozšíření do hry přidává také možnosti hraní historických scénářů s předdefinovaným rozložením plochy.

3 Požadavky pro vývoj

Pro vývoj na počátku je zapotřebí promyslet, jakým způsobem budeme realizovat následující klíčové mechanismy.

3.1 Hrací plocha

Z důvodu specifického tvaru hracího pole (šestiúhelník) je třeba si rozmyslet, jakou formou budeme hrací pole realizovat, abychom mohli detekovat hrany a body pro stavění cest a budov. Samozřejmě je důležité promyslet také způsob vykreslování plochy a zvolení nejschůdnějšího řešení. Toto rozhodnutí je závislé na použité platformě.

3.2 Uměla inteligence

V rámci vyvíjené UI[1] je důležité specifikovat, jaké konkrétní kroky bude UI[1] řešit, případně které může ignorovat, jakým způsobem se zachová při zpracovávání informací a také data, jež bude mít k dispozici - tzn. data totožná s aktivními hráči hry, nebo data za celou hru pro rozklíčování širšího kontextu (bude znát všechny informace o hráčích). Nejtěžším požadavkem z hlediska logiky je naučit UI[1] obchodovat, tzn. co je výhodné a co naopak výhodné není. Tento aspekt je velmi komplikovaně řešitelný z důvodu subjektivních pohledů vývojáře na výhodnost jednotlivých směn.

3.3 Síťová komunikace, databázová základna

Pro začátek je nutno si stanovit, na jaké bázi bude fungovat komunikace Klient-Server, a zvolit příslušnou architekturu. Toto je opět závislé na vyvíjené platformě. V rámci komunikace je třeba mít vedenu databázi informací o klientech a jejich statistiky, tzn. je nutné zamyslet se nad datovou strukturou připojené databáze.

[1] UI = umělá inteligence

4 Návrh aplikace

4.1 Platforma

- Programovací jazyk: C#
- Programovací jazyk: Microsoft .NET Framework 4

4.2 Architektura Klient/Server

Pro řešené téma se nejvíce hodí architektura Klient/Server z důvodu udržování veškerých informací o hře na serverové části. Server řeší veškeré operace, které nevyžadují přímý zásah uživatele. Základním požadavkem při řešení bylo oddělit server a klienta do struktury, v níž server provádí veškeré operace a klient pouze zobrazuje data dodaná ze serveru, případně provádí jen nejnutnější kroky, které jsou nezbytné vykonávat na klientské části (zadávání vstupů pro vyvolávané akce na serveru, např. stavění budov). Výhodou tohoto konceptu je také to, že v případě, kdy provádíme změnu, tak její implementace je ve větší míře zasazena do serverové části, tudíž klientské části se dotkne jen minimálně (záleží na druhu změny).

4.2.1 Obecný pohled na akce serveru

V rámci obecného pohledu můžeme klasifikovat akce, které se provádějí na serveru do jednotlivých bodů:

- Vygenerování hrací plochy a zajištění komunikace.
- Hození kostek a obsluha tahů.
- Na základě zadaných vstupů provést patřičné akce vyvolané na straně klienta.
- Udržování aktuálního seznamu her a uživatelů (správa uživatelů, statistik a spuštěných her).
- Vnitřní řešení obchodování (vnitřní logika vytvoření nabídky, reálné přesunutí surovin na základě požadavku).
- Správa nad balíkem speciálních karet.

Z důvodu výše popsané architektury je cílem vše centralizovat do serveru, aby byla minimální možnost ovlivnění základních mechanik klientem.

4.2.2 Obecný pohled na akce klienta

V rámci klientské části se jedná o činnost převážně pasivní, jež nezasahuje do serverové části logiky, vyjma požadavků pro provedení akcí na serveru.

- Zobrazení a aktualizace herního rozložení (vykreslování materiálů, budov, cest).
- Zobrazení a aktualizace herních výsledků (skóre, přehled surovin, speciální karty).
- Obstarání nutných akcí řešených v klientské částí (zvolení karet, které je třeba odebrat, umístění figurek Zloděj/Pirát apod.).
- Zobrazení a obstarání vnějšího procesu obchodování (jaké suroviny, způsob směny).

4.3 Detailní pohled na jednotlivé části

4.3.1 Obecný popis

Každá část řešení (klient, server) mají své specifické třídy a knihovnu, kterou společně sdílí kvůli komunikaci na základě binární serializace. Aby vše bylo korektní, je zapotřebí udržovat totožnou společnou knihovnu jak v serverové části, tak v části klientské. V případě že by tomu tak nebylo, jednotlivé části mezi sebou nedokázaly komunikovat a nastala by chyba při přenosu dat (v části při serializaci, případně deserializaci).

4.3.2 Klient

Klient je realizován WinForm aplikací. V rámci klientské části jsou nejdůležitější třídy pro vykreslování, mezi tyto třídy patří:

- main_graphics - centrální třída pro vykreslování hlavního hracího pole.
- picture_class - třída reprezentující samotný obrázek jednoho hracího pole.
- super_card_paint - třída, jež obstarává vykreslení speciálních karet(Rytíř, Pokrok atd.).
- exchange_paint - třída zastřešující vykreslování výměn mezi klienty.

Třída main_graphics je nadřazenou třídou třídy picture_class, zároveň v této podřazené třídě máme uloženy centrální informace o daném obrázku, mimo jiné je zde umístěn objekt typu PictureBox. Třída exchange_paint zastřešuje obchodní styky mezi hráči, obsahuje dvě částí. Část s nižším oprávněním a část s plným oprávněním. Na plná práva má nárok pouze hráč, jenž nabídku vytvořil (tento hráč musí mít možnost plného nastavení směny, tzn. zrušení, přijmutí vybrané odpovědi od hráčů). V případě že zainteresovaný klient je pouze příjemce, tzn. není na tahu, tak jeho přístup k nabídce je izolován pouze na přijmutí, upravení nabídky a zobrazení jejího detailu. Třída super card_paint zastřešuje obsluhu použití speciálních karet, tzn. Rytíř, Pokrok. Třídy main_graphics, exchange_paint, super_card_paint tvoří dynamicky generované objekty typu Panel, které sdružují funkční prvky dle požadované akce. V případě jakékoliv změny je tudíž možno zasáhnout pouze do dynamických generovaných komponent.

4.3.3 Server

Server je realizován jako konzolová aplikace, jelikož zde není potřeba mít grafický výstup. Serverová část obsahuje celou funkční logiku hry a správu uživatelů. V rámci serverové části můžeme třídy obsahově rozdělit do několika skupin:

- Správa klientů
- Hrací plocha
- Pravidla
- Umělá inteligence
- Obchod
- Databázová základna
- Správa hry

  - Speciální karty
  - Bonusové karty
  - Správa hráčů
- Komunikace

Detailně jsou všechny tyto bloky popsány v sekci **Implementace serveru**.

4.3.4 Společná knihovna Libary.dll

Důležitým prvkem pro celkovou komunikaci mezi klientem a serverem je společná knihovna Libary.dll, která zajišťuje konzistentnost dat při komunikaci mezi klientem a serverem. Centralizovanou třídou, která shlukuje veškerá data ze strany serveru, je třída main_send_class a ze strany klienta třída inputs (detail viz. sekce **Rozbor komunikace**). Při pohledu do obsahu knihovny zjistíme, že v porovnání se serverem je v klientské části znatelně méně tříd a to z důvodu sdílení pouze nutných tříd. Server v rámci této knihovny obsahuje ještě další potřebné třídy, bez nichž by nemohl korektně fungovat. Sdílené třídy bychom mohli klasifikovat do skupin:

- Hrací pole(detailně rozebráno v sekci **Implementace serveru**)
  - Arrays - třída zastřešující jedno hrací pole.
  - street - třída zastřešující jednu cestu v poli.
  - point - třída zastřešující jeden bod pro umístění vesnice, města v poli.
- Obchod (detailně rozebráno v sekci **Implementace serveru**)
  - exchange_main - třída zastřešující hlavní nabídky od hráčů.
  - response_change - třída zastřešující odpovědi na vyvolaný obchod.
- Hráči a statistiky (detailně rozebráno v sekci **Implementace serveru**)
  - client_statistics - třída zastřešující statistiky hráče.
  - cl_in_game - třída zastřešující výsledky hráčů v dané hře.
  - bonus_card_prm - třída zastřešující parametry jedné bonusové karty.
- Hlavní komunikace (detailně rozebráno v sekci **Rozbor komunikace**).

4.4 Komunikace

4.4.1 Obecný popis

Komunikace funguje na základě binární serializace posílané pomocí streamu. Z hlediska efektivity je toto řešení mnohem výhodnější než případná komunikace přes XML soubor nebo textovými řetězci. Výhodou tohoto řešení je hlavně minimální velikost posílaných dat v porovnání s výše zmíněnými metodami a nenáročnost celého procesu, kdy volající strana pošle objekt a strana přijímající objekt přijme a patřičně zpracuje. Podmínkou pro tento způsob komunikace je mít správně nastaven proces čtení dat, přičemž jako nejlepší řešení pro aktualizace na pozadí se jeví mít tuto funkčnost spuštěnu ve speciálním vláknu provádějícím na pozadí veškeré úkony. Jsou definovány dvě třídy, které si server a klient mezi sebou vyměňují:

- inputs

- main_send_class

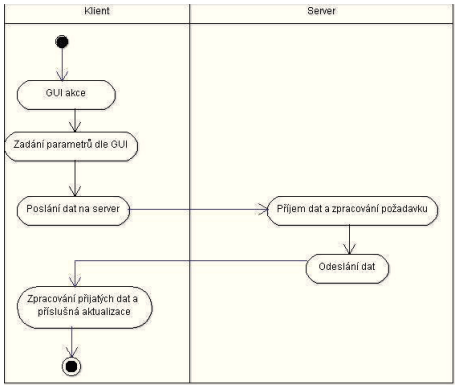
Tyto třídy se nacházejí ve společné knihovně, tudíž je zde požadavek na totožnost, aby byl přenos dat úspěšný. V případě nekonzistentnosti se nepodaří data zpracovat a dojde k vyvolání výjimky. Třída inputs je hlavním komunikačním prvkem pro klientskou část a naopak třída main_send_class je centralizovaný průřez skrz danou hru kde, jsou data, jež je nutno zpracovat v rámci klientské části.

4.4.2 Z pohledu klienta

V případě že budeme nahlížet na komunikaci z hlediska klienta, lze celkový postup shrnout do několika kroků:

- Zpracování požadavku vyvolaného GUI.
- Zapsání parametrů do objektu typu inputs.
- Odeslání objektu typu inputs na server pomocí binární serializace a TCP spojení.
- Vyčkání na odpověď a následné zpracování výsledku.

Obecnou aktivitu můžeme interpretovat pomocí níže uvedeného Activity diagramu, který nám obecně nastíní průběh při zadání požadavku.



Obrázek 2: Průběh zadání požadavku

Komunikační kanál se serverem je v této části neustále aktivní z důvodu aktualizace dat od více hráčů(klient pozná zaslání dat od serveru pouze neustálým testováním). Aby byla možná neustálá kontrola přijatých dat, běží tento proces v pozadí ve speciálním vláknu, které se stará o příjem a zpracování dat, aktualizaci a úpravy GUI.

4.4.3 Z pohledu serveru

V případě náhledu na komunikaci z hlediska serveru se nám akce rozrůstají dle typu požadované operace. V případě že se jedná o prvotní přihlášení, je zapotřebí ověřit klienta napříč DB, zdali je u nás aktuálně veden. V případě že není, server odešle obecný objekt main_send_class upravený na daný typ situace. Podobná situace nastává při vytváření, případně připojení do her, kdy opět navracíme obecný objekt daného typu v případě, že se operace nezdařila. V opačném případě je navrácen specifický objekt dle daného typu operace, tzn. pro vytvoření hry nebo připojení do existující hry se nám navrátí objekt s parametry specifickými pro danou instanci hry.

Jednotlivé kroky bychom mohli obecně standardizovat následovně:

- Vyčkávání na požadavek.
- Zpracování požadavku dle daného typu.
- Zaslání výsledného objektu konkrétnímu uživateli, případně celé skupině.

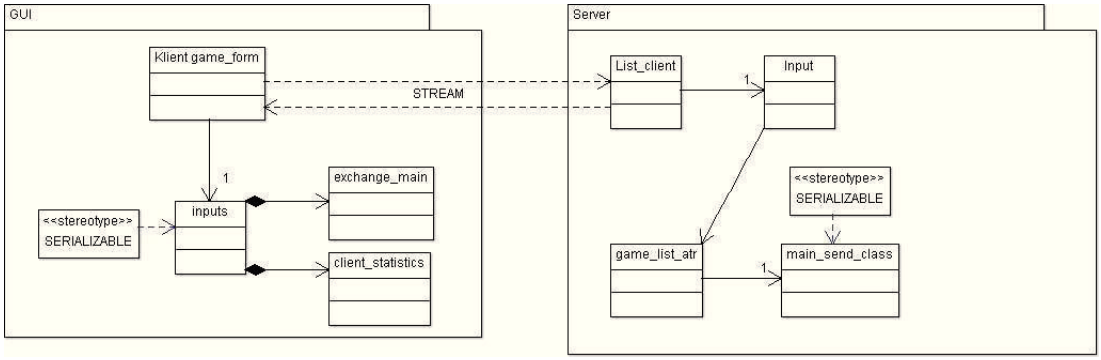
V tomto typu hry můžeme rozlišovat dva typy distribuce dat - data, která posílám pouze jednomu hráči, a data posílaná celé skupině hráčů. Typickým představitelem první skupiny mohou být např.seznamy her, které se posílají pouze na volajícího hráče. Představitelem druhé skupiny je např. postavení budovy některým z hráčů ve hře, kdy tato změna se musí zobrazit všem zainteresovaným hráčům ve hře.

Pro korektní zpracování požadavků na serverové části je nutné aplikovat stejnou logiku jako v předchozím bodě, tzn. je zapotřebí mít spuštěna externí vlákna, jež mají na starosti detekci příchozích zpráv. Důležité je mít korektně zpracovánu správu vláken, aby nedošlo k poslání dat jinému klientovi než bylo původně zamýšleno.

5 Implementace serveru

5.1 Rozbor komunikace

Komunikaci mezi klientskou částí a serverem můžeme znázornit třídním diagramem:



Obrázek 3: Diagram znázorňující komunikaci

5.1.1 Komunikační třída inputs

Jak už bylo v předchozích kapitolách zmíněno, pro zpracování požadavku na serveru slouží hlavní třída inputs, kde jsou deklarovány všechny potřebné parametry pro provedení akce. Proměnné třídy můžeme opět rozdělit do několika skupin:

- Stavba a přesun objektů
- Obchod
- Přihlašovací údaje

Přičemž mezi nejdůležitější parametry této třídy bych zařadil tyto proměnné:

- choice - reprezentuje akci, kterou chce hráč provést a je reprezentován datovým typem string.
- index_in_field - jednoznačný identifikátor umístění objektu v jednom hracím políčku.
- value_field - jednoznačný identifikátor hracího políčka na hrací ploše.
- number_game - jednoznačný identifikátor hry.

Jednotlivé proměnné ovšem v některých případech mohou být použity také pro jinou podobnou akci z důvodu zbytečného vytváření totožných proměnných.

5.1.2 Komunikační třída main_send_class

Jak bylo výše zmíněno, tato třída obsahuje konsolidovaný průřez vlastnostmi hrané hry. Mezi klíčové proměnné mimo jiné patří:

- status_operation - vyjadřuje, zdali požadavek poslaný serveru proběhl úspěšně.
- game_ar - obsahuje kompletní herní pole se všemi budovami a objekty.

- game_box - v případě aktualizace seznamu her toto pole obsahuje celkový seznam aktivních her.
- clients_gm - seznam hráčů, již jsou přítomni v hrané hře.
- client_in_play - indikace hráče, který je právě na tahu.
- change_mn - obsahuje informace o vytvořeném obchodu mezi hráči.

Mimo jiné obsahuje tato třída také výsledkové kódy operací, jejichž interpretace je následně zobrazena hráči ve formě textu. Tyto výsledkové kódy rozlišujeme dva: operation_text a error_txt. Jak je z názvu patrné, první proměnná je použita jako informační zdroj a druhá jako výsledkový kód v případě chyby. Mimo jiné tato třída také obsahuje prvopočáteční proměnné při tvorbě hry, tzn.game_number reprezentující jednoznačné ID hry reprezentované datovým typem string, a tým, který byl volajícímu hráči přiřazen (index_user).

5.2 Zpracovávání požadavku, rozbor základní třídy Input

V případě zaslání objektu na server je tato akce okamžitě rozpoznána a server reaguje na zadanou akci uloženou v parametru choice tím, že celý objekt typu inputs, který obdržel, předá objektu typu Input.

Tento objekt na základě zadaného parametru zpracuje požadavek. Rozlišujeme tyto druhy požadavků:

- "login"- Zaštiťuje přihlášení na server a zajištění ověření klienta v databázi.
- "join"- Připojení do existující hry.
- "create_g"- Vytvoření hry dle zvolených požadavků (Námořníci, základní pravidla, předvolená mapa).
- "end_move"- Ukončení tahu.
- "move_thief"- Přesunutí zloděje do předem předdefinované lokace.
- "build"- Stavba objektů, rozřazení na daný typ objektu se provádí až na úrovni logiky konkrétní instance hry.
- "exchange_server"- Obchodování se serverem (přístavy,základní kurz).
- "exchange_client"- Obchodování s hráči. Tento požadavek je spjat s požadavkem exchange_resp.
- "exchange_resp"- Vložení odpovědi od hráčů na vytvořenou nabídku.
- "refresch_list"- Vytvoření seznamu her a jejich obsazenost.
- "use_special_card"- Použití speciálních karet (Rytíř, Pokrok atd.).
- "buy_special_card"- Koupení speciální karty.
- "selected_card_down"- Odstranění vybraného balíku karet.
- "move_boat"- Přesunutí lodě na jinou destinaci (Námořníci).
- "golden_river"- Ošetření v případě přítomných karet "Zlatá řeka"(Námořníci).

- "change_pass"- Změna hesla na účtu.
- "remove_offer"- Odebrání nabídky, kterou klient vytvořil pro obchod s hráči. Tento požadavek je spjat s exchange_client.

Každá z těchto položek je ošetřena a má své specifické parametry, které získává s klientem poslaného objektu. Třída Input má tři veřejné metody - metodu pro obsluhu vstupů, metodu pro obsluhu vyřazení klienta při nečinnosti nebo odpojení a metodu pro obsluhu umělé inteligence, která je rozebrána v níže uvedené části. Zbytek metod pro ošetření je pouze privátní, tudíž do nich můžeme přistupovat pouze v rámci této třídy. Všechny tyto tři veřejné metody jsou uzamknuty klíčovým slovem **lock**. Je to z toho důvodu, aby v jednu chvíli do těchto bloků kódu mohlo přistoupit pouze jedno vlákno a tudíž aby nedošlo ke změně dat více uživateli naráz, což by mělo za následek nekonzistentnost dat napříč klienty stejné hry.

5.3 Vytvoření, připojení do hry

5.3.1 Obnovení listu her

V případě, kdy klient pošle požadavek pro zjištění aktuálního seznamu probíhajících her, je zapotřebí vygenerovat korektní data z aktuálního seznamu probíhajících her. V rámci třídy Game_list je vytvořen list všech her. Metoda pro vytvoření tohoto seznamu vytvoří nové pole typu string o třech sloupcích, kde postupným procházením listu her pomocí bloku foreach vyřadí všechny neaktivní hry a do výsledného pole vloží ID hry, typ pravidel a počet hráčů ve hře ve formátu *počet připojených/počet potřebných*. Následně tento seznam vrátí zpět volajícímu objektu, který jej odešle zpět klientu žádajícímu o tento seznam.

5.3.2 Vytvoření hry

Pokud je detekován požadavek na vytvoření nové hry a vyvolána patřičná metoda v hlavní třídě zpracování požadavku Input, zavolá se metoda v rámci třídy Game_list pro vytvoření hry, které předáme jako parametr data o zakládajícím klientovi (stream, ID uživatele), typ vytvářené hry (1 - základní pravidla, 2 - rozšíření Námořníci), a nakonec v případě vytvoření hry pomocí předdefinovaného rozložení číslo scénáře, dle kterého chceme hrací pole vygenerovat. V případě že negenerujeme hru dle předem vytvořeného rozložení, má tento parametr hodnotu *"none"*. Bližší specifikace vytvoření hracího rozložení je popsána v části **Herní plocha**.

5.3.3 Vnitřní struktura hry

Při vytváření hry se automaticky vygeneruje i objekt, jež reprezentuje stav a veškeré nastavení hry. Tento objekt je instancí třídy game_list_atr sdružující veškeré metody a přístupy pro změnu v daném objektu. Hlavními proměnnými této třídy jsou:

- clients_gm - instance třídy cl_in_game, rozebrána v sekci **Klient a jeho vlastnosti**
- rl1, rl2 - pravidla pro daný typ hry, instance třídy Rules(1,2), rozebrána v sekci **Pravidla**
- team_in_play - proměnná určující aktuální tým na řadě
- cnt_street, cnt_village, count_street, count_village - proměnné pro stavění volných vesnic a cest, rozebrána v sekci **Počátek hry**
- road_cnt - instance typu longest_road, slouží pro výpočet atributu nejdelší cesta, rozebrána v sekci **Nejdelší cesta**

- main_spec_card - instance typu special_card - má pod správou distribuci speciálních karet (Rytíř, Pokrok atd.), rozebrána v sekci **Speciální karty**
- exch_main - instance typu exchange_main jenž zastřešuje výměnný obchod se serverem nebo hráči, rozebrána v sekci **Obchodování**
- cube_n - proměnná typu int reprezentující hodnotu hodu kostek

Při vytvoření nebo připojení do hry je klientovi, jenž vyvolal tuto akci, zaslána také proměnná index_user neprezentující jeho pořadí v poli klientů. Hodnota této proměnná je klíčová při komunikaci s danou hrou, jelikož na základě této hodnoty se dohledávají a upravují veškerá hráčská data v dané hře.

5.3.4 Připojení do existující hry

V případě požadavku na připojení do existující hry se jako parametr potřebné funkci předá ID hry, které bylo dle jejího názvu dohledáno v listu her, a stream hráče, jež se chce do hry připojit. Pokud vše proběhne korektně, hráčovy parametry se přidají do aktuální hry, a to konkrétně do objektu typu cl_in_game, který zastřešuje hráče hrajícího danou hru. V případě že je hra již zaplněna, tudíž není žádná volná pozice, volajícímu hráči je vrácen objekt se statusem **false** a patřičným výsledkovým kódem. V případě úspěšného připojení ke hře volající hráč obratem obdrží obnovený objekt typu main_send_class obsahující patřičné parametry hry, jež byly popsány v předešlých sekcích. Tato změna se samozřejmě projeví také u všech hráčů, kteří jsou připojeni v této hře.

5.4 Struktura tahu

5.4.1 Počátek hry

Aby byly splněny specifické podmínky pro počátek hry (počáteční rozložení budov zdarma, s upravenými pravidly pro lokaci), je zapotřebí mít vedeny aktuální počty postavených objektů na počátku tahu. K tomuto účelu nám slouží proměnné cnt_village, cnt_street, count_street, count_village obsažené ve třídě game_list_atr, v nichž uchováváme počáteční stavy a aktuální stavy objektů. Při následném budování jednotlivých objektů jednoduše pouze porovnáme počty na počátku tahu a aktuální počty a dle výsledku můžeme stanovit následující postup. V tomto případě je nutno postavit zadaný počet objektů, jinak není možné ve hře pokračovat. Obdobnou metodiku lze použít při použití speciálních karet pouze v modifikované podobě (detail v sekci **Speciální karty**). Počáteční postavení hry je identifikováno metodou **verification_begin()** nabývající hodnot true nebo false, kterou mohu logicky odvodit. V případě že musí proběhnout dvě kola počátečních tahů, pak tah, který následuje po těchto tazích, již nemůžeme klasifikovat jako tah počáteční. Z toho vyplývá, že počátek hry je v prvních dvou kolech hry. Jednoduchým přepočtením dostaneme nutný počet tahů, což je 8 v případě, že máme 4 hráče.

5.4.2 Standardní tah

Jak bylo popsáno v sekci Pravidla a jejich rozšíření, tah si z logického hlediska můžeme rozdělit do několika bodů. Tyto jednotlivé celky lze následně jednoduše implementovat a separovat od sebe, aby nebyly v jedné funkci. Standardní tah z hlediska logiky serveru probíhá po přičtení surovin a hodu kostkou. Tyto operace byly provedeny v předchozím tahu, jakmile byla vyvolána metoda pro ukončení tahu, tudíž hráč na tahu nyní obdrží datový objekt, jehož obsah

vyvolá patřičné odezvy např. vizualizace strhnutí surovin v případě hození čísla 7 při nesplnění podmínek atd.

V případě kdy není vyžadována žádná interakce od hráče, může hráč ihned stavět objekty a provádět povolené akce; ovšem na celý tah má časový limit 300 sekund, přičemž uplyne-li tento časový limit dřív, než je úspěšně ukončen tah, klient je automaticky odebrán ze hry a nahrazen umělou inteligencí.

5.4.3 Ukončení tahu

Metodiku pro ukončení můžeme definovat jednoduchým vývojovým diagramem (Příloha č.1). Z diagramu je patrné že průběh ukončování ovlivňují několik situací:

- Počátek hry (postavení povinných budov detail v sekci **Počátek hry**)
- Výhra
- Nutné povinné akce v tahu (strhnutí materiálů hráčům za splnění určených podmínek, detail v sekci **Zloděj/Pirát**)

5.5 Hráč a jeho vlastnosti ve hře

Vlastnosti hráče v právě probíhající hře jsou uloženy v instanci třídy cl_in_game. Standardně jsou v této třídě umístěny mimo jiné tyto hlavní atributy:

- card_l - přehled získaných surovin
- bonus_l - přehled speciálních karet (blíže rozebráno v sekci **Speciální karty**)
- change_l - v případě postavení přístavu se zde odblokuje patřičný výměnný poměr
- score - aktuální počet vítězných bodů hráče
- login_user - uživatelské jméno hráče
- longest_road, maximum_army - identifikátory zdali hráč získal bonusové karty
- number_7, golden_river - identifikátory v případě, že je zapotřebí provést povinné operace
- access_build - pole reprezentující požadavky na povinné objekty pro stavbu

Podrobněji rozebereme pouze některé části. Proměnná change_l zastřešuje povolené kurzy pro výměnu se serverem. Na počátku jsou všechny tyto kurzy zakázány až na jeden, a to kurz základní (4:1). V případě postavení objektu typu vesnice či město na místě, na němž je postaven přístav, se hráči automaticky také odblokuje daný kurz výměny.

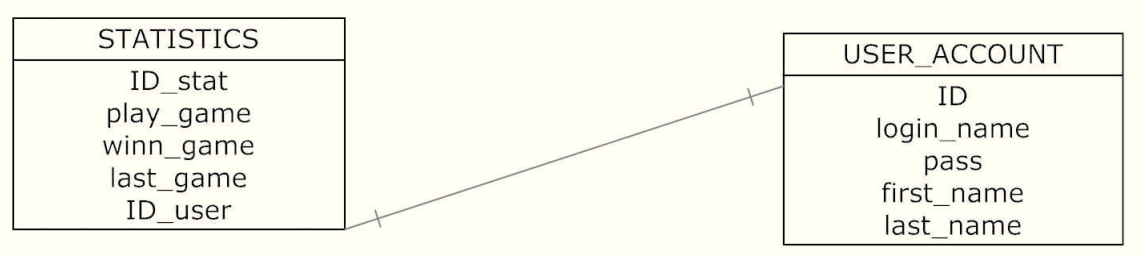
Proměnná acces_build je reprezentována jednorozměrným polem, kde index prvku nám určuje objekt zájmu (město, vesnice, cesta) a hodnota prvku definuje požadovaný počet stavěných objektů na tah.

Velmi důležitou funkci také plní proměnné number_7 a golden_river. Díky těmto proměnným identifikujeme, zdali uživatelé provedli povinné akce (povinné odebrání surovin, povinné přidání surovin). Ověření probíhá při požadavku o ukončení tahu. Aby se tah ukončil, oba tyto parametry musí nabývat hodnoty false.

5.6 Databázová základna a statistiky

5.6.1 Obecný popis

Databázovou základnu tvoří dvě tabulky STATISTICS a USER_ACCOUNT. Jejich vzájemné propojení společně s atributy je znázorněno v následujícím E-R Diagramu.



Obrázek 4: ER Diagram

Dle obrázku vyplývá, že každý záznam obsažený v tabulce USER_ACCOUNT má zároveň záznam v tabulce STATISTICS, tzn. každý uživatel má právě jeden záznam v tabulce STATISTICS. Jednotlivé editace záznamů jsou prováděny pomocí předdefinovaných funkcí a triggerů. Přehled funkcí a triggerů:

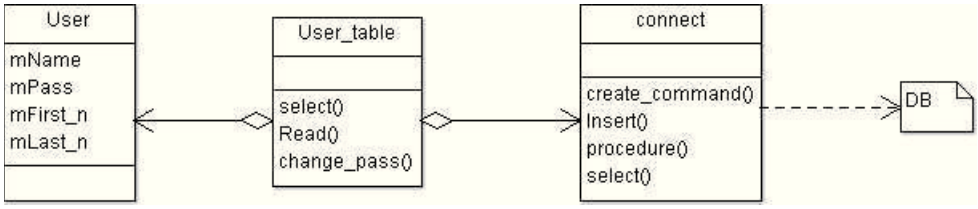
- game_up - Hráči určenému loginem zvýší atribut PLAY_GAME o jedna a atribut LAST_GAME nastaví na aktuální datum a čas (serveru).
- winn_up - Hráči určenému loginem zvýší atribut WINN_GAME o jedna.
- change_pass - Hráči určenému loginem změní heslo del zadaného paramtru.
- INSERT_USER - V rámci vkládání nového uživatele je automaticky vytvořen také záznam v tabulce STATISTICS.

5.6.2 Implementace

Pro implementaci je aplikován návrhový vzor Data Mapper(reprezentuje použití techniky ORM), který zastřešuje komunikaci a práci s databází. Obsahuje tyto třídy:

- connect
- statistics
- statistics_table
- user
- user_table

Třídy statistics_table a user_table reprezentují přístupové metody pro práci s daty, tzn. v případě kdy chceme provést operaci, jež zasahuje do konkrétní tabulky databáze,vyvolá se metoda této třídy, pomocí níž se následně předají patřičné parametry třídě connect. Proměnné tříd statistics a user jsou odrazem atributů tabulek v databázi. Tyto třídy mimo jiné obsahují metody pro úpravu parametrů. Hlavní třídou je třída connect, která zastřešuje fyzické spojení s databází a vyvolání požadovaných příkazů. Pro názornost je komunikace s databází znázorněna v třídním diagramu:

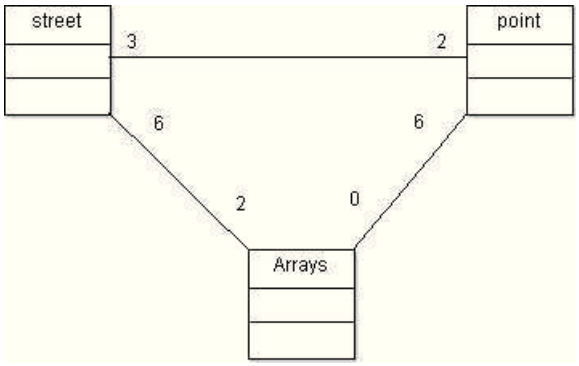


Obrázek 5: Návrhový vzor Data Mapper

5.7 Herní plocha

5.7.1 Vytvoření a propojení

Herní plochu můžeme rozdělit na předem známý počet hracích polí. Jedno hrací pole dle obecné definice obsahuje 6 cest a 6 bodů pro stavbu objektů. Na základě této premisy jsem vytvořil jednorozměrné pole typu Arrays, kde každý prvek pole reprezentuje jedno hrací pole jež obsahuje dvě pole typu street a point o 6 prvcích. Propojení jednotlivých prvků je reprezentováno třídním diagramem:



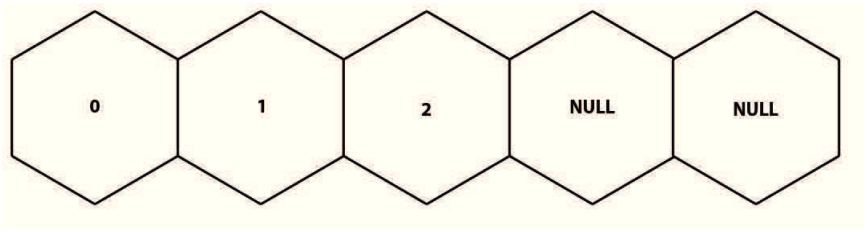
Obrázek 6: Třídní diagram herní plochy

Z hlediska hry jsou důležité pro zmiňované třídy mimo výše zmíněných tyto proměnné:

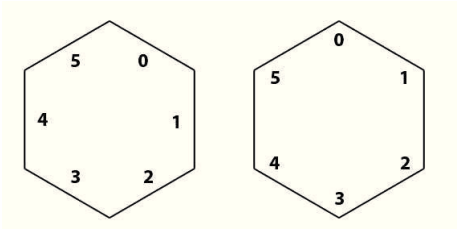
- Arrays
  - material - představuje hodnotu materiálu, jež je na tomto poli uložen
  - meassure - hodnota pole při hodu kostkami
  - thief - indikace umístění Zloděje (nabývá hodnot true nebo false)
  - pirat - indikace umístění Piráta (nabývá hodnot true nebo false)
- street
  - team - proměnná s uloženou hodnotou týmu, která má zde umístěnu cestu či loď
  - type_road - rozlišení druhu objektu (loď, cesta)
- point
  - type_point - proměnná určující typ vloženého objektu (město, vesnice)
  - team - proměnná s uloženou hodnotou týmu

– harbor - určuje, zdali je tento bod zároveň přístav
– type_harbor - stanovuje typ přístavu (obecný, speciální)
– hrb_meassure - v případě speciálního přístavu určuje materiál pro směnu

Pří pohledu na herní plochu na první pohled zjistíme, že počet hracích polí je v každé řadě nerovnoměrný. Tento rozdíl mezi řadami je v poli reprezentován hodnotou null v nepoužitých prvcích.



Obrázek 7: Indexace v poli



Obrázek 8: Umístění cest a bodů

Na obrázku můžeme vidět také číslování jednotlivých bodů a cest, které reprezentuje jednoznačné umístění daného objektu v hracím poli.

Instance objektů se mezi sebou propojují pomocí kódu o jednotné struktuře. Pro korektní průběh je zapotřebí nastavit počáteční dvě proměnné max_in_row, basic_in_row. První proměnná definuje minimální počet hracích polí v herním plánu a druhá naopak maximální počet hracích polí v jedné řadě. Dle těchto parametrů postupným přičítáním a odečítáním (herní plocha má tvar šestiúhelníku, tudíž v první polovině budeme vždy přičítat číslo 1 a v druhé naopak odečítat) jsme schopni navzájem mezi sebou propojit jednotlivé objekty (cesty, body, hrací pole) a tak docílit efektu, kdy změníme objekt v jednom umístění. Tato změna se projeví také v ostatních sdílených objektech (např. cesta je sdílena mezi dvěma hracími poli, změna se projeví v obou polích pouze na jiných umístěních).

Pro popisovanou problematiku jsou implementovány metody:

- set_instances() - vytvoření počátečních instancí v poli
- set_street() - vzájemné propojení cest
- set_point() - vzájemné propojení bodů
- set_streets_point() - propojení cest a bodů mezi sebou

5.7.2 Generování materiálů

Každé hrací pole obsahuje parametr meassure kde je uložena hodnota materiálu. Tento parametr může nabývat hodnot:

- 0 - Les
- 1 - Pastvina
- 2 - Pole
- 3 - Pahorkatina
- 4 - Hory
- 5 - Poušť
- 6 - Zlatá řeka (rozšíření Námořníci)
- 7 - Moře (rozšíření Námořníci)

Generování materiálu je založeno na bázi náhodně generovaného čísla pomocí proměnné typu Random. Hodnoty mohou být pouze z předem definovaného rozsahu (základní pravidla 0 - 5, rozšíření Námořníci 0 - 7).

Pro vygenerování a přiřazení materiálů hracím polím slouží metoda set_material(), jež na počátku vytvoří jednorozměrné pole s hodnotami reprezentující počet dané suroviny v herní plánu, identifikací daného typu materiálu je index prvku v poli. Následně projde celé herní pole a prvkům, které nenabývají hodnot null, přiřadí náhodně generovaný materiál. Při každém přiřazení materiálu hracímu poli odečte z kontrolního pole za daný materiál číslo 1.

5.8 Pravidla

V rámci logiky serveru existují dvě třídy s názvem Rules1 a Rules2. Jedná se o třídy deklarující konkrétní pravidla hrané verze hry (Rules1 = základní pravidla, Rules2 = rozšíření Námořníci). Tyto třídy obsahují deklaraci hracího pole a metody popisované v sekci **Herní plocha**. Mimo zmiňovaných metod jsou zde také metody zastřešující stavby budov a ověřovací podmínky pro korektní stavbu (detailně rozebráno v sekci **Stavba objektů**).

5.9 Obchodování

Pro realizaci obchodu jsou důležité dvě třídy:

- exchange_main
- response_change

Rozlišujeme dvě úrovně obchodu (server, hráč). V případě první možnosti proběhne jednoduchá výměna surovin na základě odblokovaných přístavů hráče (viz sekce **Hráč a jeho vlastnosti ve hře**); operace se nemusí zdařit z důvodu nedostatku surovin pro směnu na straně klienta.

Způsob výměny surovin s hráčem již tak přímočarý není. Klient, jenž založil výměnu, vytvoří nový objekt typu exchange_main, přičemž jako základní parametry je nutno konstruktoru předat parametry:

- "Co potřebuji"- jednorozměrné pole *need*
- "Co za to nabízím"- jednorozměrné pole *offer*

Jakmile je výměna vytvořena, ihned se data distribuují na všechny aktivní hráče ve hře, kteří v případě zájmu vrátí zpět objekt typu response_change, udávající parametry:

- team - identifikátor hráče, jenž nabídku akceptoval/upravil
- offer - pole surovin, za něž klient provede výměnu

Tento objekt je následně přidán do hlavního objektu zastřešující celou výměnu. Po takovéto výměně tedy objekt zastřešující celou výměnu obsahuje základní parametry od zakládajícího hráče a seznam nabídek od hráčů, kteří na nabídku reagovali.

5.10 Stavba objektů

Jak bylo již zmíněno v sekci Pravidla a jejich rozšíření, rozlišujeme tři typy budov a jejich identifikátor:

- 1 - vesnice
- 2 - město
- 0 - cesta
- 3 - loď

Obecně můžeme postup při stavbě shrnout do několika kroků:

- Vyvolání stavby objektu klientem.
- Vyvolání metody pro stavbu v hlavním obslužném objektu typu Input.
- Vyvolání metody pro ověření konkrétního stavu hry v instanci typu game_list_atr dané hry (počátek hry, použití speciálních karet).
- V případě splnění podmínek vyvolání stavby v objektu reprezentující pravidla (datový typ Rules1, Rules2).
- Dle výsledkových kódů proběhne komunikace s hráči.

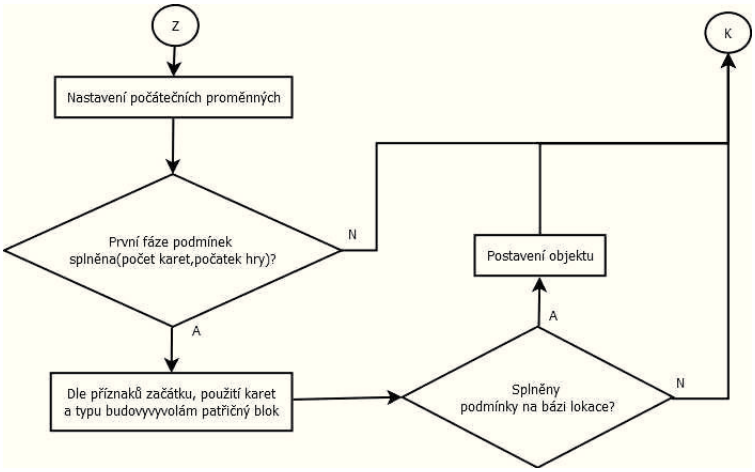
K vyvolání stavby budov dojde v případě klientského požadavku (parametr choice třídy inputs má hodnotu "*build*"). Nejdůležitějšími prvky, jichž je zapotřebí, aby hráč vyplnil při požadavku na stavbu, jsou:

- Typ budovy, kterou chceme postavit (viz. počátek této sekce).
- Index hracího pole, na které chce objekt umístit.
- Index, jež reprezentuje umístění ve zvoleném hracím poli.

V případě korektních dat je vyvolána metoda insert_obj v hlavním obslužném objektu typu Input. Ten vyvolá metodu pro stavění budovy v konkrétní hře (instanci třídy game_list_atr) a předá této metodě parametry pro stavbu. Vyvolaná metoda obstarává celou operaci stavby, tzn. řeší, zdali se jedná o počáteční tahy, v případě různých typů hraných pravidel patřičně reaguje. V rámci prvotního ověřování je zapotřebí ověřovat, zda se jedná o počátek hry, použití speciálních karet, jelikož se jedná o specifické stavy při stavbě objektů. Počátek hry a užití speciálních karet je specifické ve stavěných objektech bez nutnosti strhnutí surovin za daný typ objektu.

Dle pravidel se jedná v případě počátku hry o dvě vesnice a dvě cesty. V případě akčních karet hovoříme o speciální kartě, jež dovoluje volné postavení dvou cest.

V případě že požadavek na stavbu objektu projde prvním ověřením tzn. postoupí pouze v případě, kdy se jedná o počátek hry nebo hráč používá speciální kartu, případně má dostatek surovin pro stavbu daného typu objektu, je vyvolána finální metoda stavby, která provede finální ověření na základě lokace umisťovaného objektu. Dle typu pravidel musí objekty splňovat podmínky uvedeny v sekci **Pravidla a jejich rozšíření**. Při obecnějším pohledu na zpracování můžeme vyjádřit stavbu objektů vývojovým diagramem:



Obrázek 9: Vývojový diagram stavby objektů

5.11 Speciální karty

Dle pravidel rozlišujeme tři druhy speciálních karet. Na počátku hry je vždy balík speciálních karet, který má přesně definovaný počet karet za konkrétní druh. Pro speciální karty na serveru nalezneme třídy:

- special_card
- bonus_card_prm

První z uvedených zastřešuje balík karet tzn. třída obsahuje list objektů typu bonus_card_prm. Druhá zmíněná třída reprezentuje konkrétní speciální kartu a její vlastnosti:

- card_type - rozlišujeme tři druhy karet (Bod - 1, Rytíř - 2, Pokrok - 3)
- card_use - indikátor, zda je karta již použitá (hraje roli v případě karty Rytíř)
- card_enabled - vymezuje použitelnost karty (v tahu, kdy byla karta získána, kartu nelze použít)
- card_name - určuje název karty
- in_list - identifikátor, zda se karta nachází v balíku, či nikoliv
- ID_card - unikátní identifikátor karty

V případě požadavku klienta o koupi speciální karty se v první řadě při distribuci ověří, jestli již balík není prázdný; pokud by nastala tato situace, navrátí metoda hodnotu null. V jiném případě hra nalezne kartu, která je aktuálně v balíku na základě náhodného čísla, abychom zachovali poměr náhody ve hře. V případě úspěchu výsledný objekt typu bonus_card_prm je vložen do seznamu speciálních karet v objektu zastřešujícím hráče ve hře.

5.12 Nejdelší cesta

Dle pravidel rozlišujeme dva principy nejdelší cesty:

- Dle základních pravidel (pouze cesty)
- Dle pravidel rozšíření Námořníci (speciální případy kombinací lodí a cest)

Z obecného hlediska se jedná o hledání maximální cesty v grafu. Pro tuto metodiku je použita rekurze (metoda volá sama sebe).
Základními předpoklady pro danou implementaci hry jsou:

- Každá cesta/loď má dva objekty typu point
- Každý objekt typu point obsahuje tři objekty typu street

Dle těchto premis můžeme určit teoretickou metodu pro dohledání nejdelší cesty v grafu. Obecně můžeme metodu postupným porovnáváním, kdy porovnáme dvě cesty ze tří v daném bodu (vyloučíme zdrojovou cestu ze, které vcházíme do daného bodu, zároveň platí že každý bod má minimálně dvě a maximálně tři cesty). V případě že zde hráč na dané lokaci nemá cestu vrátíme číslo 0 v druhé případě zavoláme rekurzivně metodu pro výpočet nejdelší cesty a k výsledku přičteme číslo 1.

V praktické implementaci je vytvořen List již navštívených objektů typu street vůči kterému každá rekurzivně volání metoda ověřuje zda-li objekt typu street který chce nyní analyzovat se nenachází v tomto listu tzn. v případě pozitivního nálezu vracím ihned hodnotu 0, jelikož objekt už byl jednou porovnáván. Oproti teoretické metodě zde porovnáváme všechny 3 cesty (nelze určit, která cesta je zdrojová dle názvu proměnné, v případě porovnávání zdrojové cesty však bude vrácena vždy hodnota 0, jelikož se tento objekt už bude nacházet v listu navštívených objektů typu street) objektu typu point. Metoda následně vrátí největší počet segmentů z analyzovaných třech směrů.

Metodu pro výpočet nejdelší cesty v grafu vyvoláváme vždy po úspěšném vybudování jakéhokoliv objektu typu street. Jedině tak dosáhneme aktuálních výsledků.

5.13 Největší armáda

Největší armáda je obecně vzato bonusová karta, kterou obdrží hráč, jenž má nejvíce použitých rytířů. Testovací podmínka tedy probíhá nad objektem zastřešujícím data hráčů, kdy jsou porovnávány součty z listu speciálních karet těch položek, které mají card_type roven číslu 2 (Rytíř) a zároveň tato karta má parametr card_use roven hodnotě true (příznak, že karta byla již použita).

Metoda pro ověření získání tohoto benefitu (+2 výherní body) je vyvolávána při úspěšném použití karty Rytíř.

5.14 Zloděj/Pirát

Poloha Piráta a Zloděje implementována pomocí příznaků (thief, pirat) datového typu bool. Tyto příznaky jsou součástí každého hracího pole, tzn. jejich deklaraci obsahuje třída Arrays. V případě kdy je identifikována akce ze strany klienta na přesunutí Zloděje/Piráta, je na zadané umístění umístěn Zloděj/Pirát (patřičný příznak změní hodnotu z false na true). Příznak na původním umístění Piráta/Zloděje je upraven na výchozí hodnotu false.

Rozlišujeme mezi několika variantami vyvolání Zloděje/Piráta:

- Padlo číslo 7
- Použití speciální karty

V případě první možnosti navíc kromě přesunutí Zloděje na patřičnou lokaci také náhodně odebereme surovinu předem vybranému hráči, jenž má město či vesnici na bodech této nové lokace. V případě druhé možnosti zůstáváme pouze u kroku přesunutí Zloděje/Piráta. Pravidlo o odebrání surovin některému z hráčů platí pouze pro figurku Zloděj.

V případě padnutí čísla 7 je provedena také kontrola počtu surovin u všech hráčů. V případě že některý hráč má větší surovin než 7, je nucen odebrat polovinu všech surovin. Server danému hráči nastaví příznak number_7 na true, což na klientské části vyvolá patřičnou odezvu (detail v sekci **Implementace klienta**). Pro následné stržení je poté použita metoda selected_card_down(), jež je deklarována v třídě game_list_atr. Zmíněná metoda provede stržení zadaných surovin dle předaného parametru typu pole obsahující počty surovin, již budou odebírány.

V případě základních pravidel není nutná kontrola nové lokace Zloděje. Ovšem v případě rozšíření Námořníci je nutné zavést kontrolu, zda cílová lokace Piráta/Zloděje odpovídá danému typu, tzn. Piráta lze umístit pouze na materiál moře a naopak Zloděje lze umístit na každý druh materiálu vyjma moře.

5.15 Předdefinované herní plány

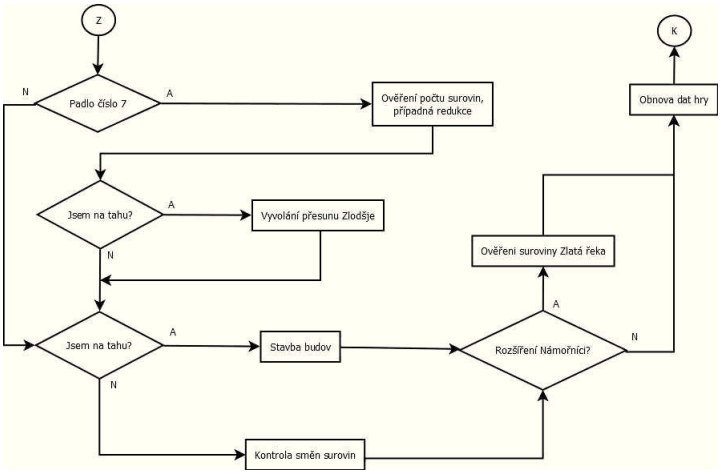
Předem připravené rozložení hrací plochy včetně čísel reprezentujících hodnotu kostek a přednastavenými přístavy jsou vytvořeny za pomocí binární serializace směřované do binárního souboru. Výsledný soubor je uložen na serveru, odkud se následně data pro vytvoření herního pole v případě zvolení scénáře čerpají. Generátor map můžeme logicky složit ze základních tříd serveru:

- Hrací plocha - Arrays, street, point
- Pravidla pro danou hru - Rules1, Rules2

Následně postačí pouze adekvátně upravit metody třídy pravidel, tzn. přidat binární serializaci, odstranit automatické vyplňování materiálů a hodnot hracích polí. Posledním krokem je vyplnit parametry hracích polí dle požadavků (meassure, material, harbor). Výsledný soubor poté při deserializaci stačí přetypovat na jednorozměrné pole typu Arrays.

5.16 Umělá inteligence

Navrhovaná logika pro realizaci hráče ovládaného počítačem částečně koresponduje s logikou reálného hráče. Navržená a implementovaná metodika neřeší vytváření výměny surovin s ostatními hráči. V případě již vytvořené nabídky je 70% šance na příjmutí této směny. Logiku chování můžeme jednoznačně vyjádřit následujícím vývojovým diagramem:



Obrázek 10: Posloupnost chování

5.16.1 UI a stavba objektů

Rozlišujeme dvě různé metodiky staveb pro počátek hry a standardní tahy. Metodika stavby objektů pro počátek hry je postavena na modelu nejvýhodnějšího neobsazeného bodu (objektu typu point). Tato výhodnost je definována číslem, jež se skládá ze tří částí. Berme v potaz, že každý bod je sdílen třemi herními poli, kde každé má svoji hodnotu pro hod kostkami. Tyto hodnoty polí (čísla 2 - 12 mimo číslo 7) mají svou váhu vypočtenou na základě permutací dvou stejných čísel o hodnotě 6. Počty jednotlivých permutací můžeme vidět zde:

Hodnota kostek	Počet permutací
2	1
3	2
4	3
5	4
6	5
8	5
9	4
10	3
11	2
12	1

Celková váha jednotlivých bodů je tedy definována součtem počtu permutací za jednotlivá hrací pole obsahující čísla kostek. Veškeré informace o daném objektu, tzn. váha, číslo hracího pole, index lokace v hracím poli jsou reprezentovány instancí třídy point_weight. Nejjednodušší metodou je výsledný List instancí třídy point_weight seřadit sestupně podle vypočtené váhy. Díky takto seřazenému seznamu můžeme začít stavět povinné budovy, aniž bychom museli zbytečně porovnávat a vyhledávat bod s největší vahou.

Metodika pro stavbu v běžném tahu je realizována následujícím pořadím stavby objektů:

- Město
- Vesnice
- Cesta,loď

Z následujícího pořadí vyplývají priority staveb (město - maximální priorita, cesta - nejmenší priorita), tzn. pokud by se UI[1] měla rozhodovat mezi stavbou města, vesnice a cesty, vždy zvolí prvek dle nejvyšší priority. Postavení vesnice je řešené podobně jako v případě výše popisované stavby na počátku hry, avšak jsou zde navíc přidány podmínky určující kritéria daná pravidly (pravidlo dvou cest, v okolních bodech nesmí být žádné budovy atd.). V případě podrobnějšího pohledu na stavění cest a lodí můžeme specifikovat základní premisu. **Vždy musí být v analyzované lokaci minimálně jedna prázdná lokace pro objekt typu cesta či loď a minimálně jedna lokace obsahující objekt typu cesta či loď patřící týmu daného hráče.** Následný postup stavby je řešen náhodným výběrem prvku v Listu všech použitelných lokací.

5.16.2 UI řešení Zloděje

Algoritmus pro umístění zloděje je založen na bázi blokování nejnebezpečnějšího hráče z pohledu výherních bodů. Jakmile klient najde tohoto hráče, zanalyzuje hrací pole, na nichž daný hráč má postaveny budovy(mimo hracích polí, kde jsou postaveny vlastní objekty hráče na tahu). Po analyzování zjistí hrací pole, jehož hodnotu lze pomocí kostek dosáhnout nejvíce permutacemi, a na toto hrací pole umístí příznak Zloděje.

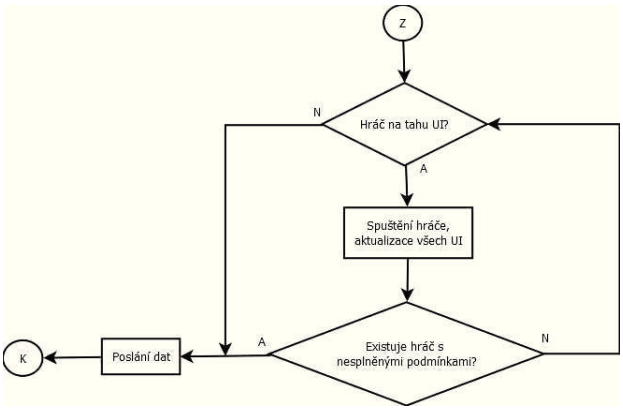
5.16.3 Implementace do běžícího procesu

Mezi hlavní třídy ovládání patří:

- UI_main - Hlavní řídící třída, obsahuje celou logiku UI hráče.
- point_weight - Třída definující vlastnosti pro stavbu objektu, klíčovým atributem je zde proměnná **weight** popsána v předešlých sekcích.

V případě kdy je hráč na ta tahu řízen umělou inteligencí, je vyvolána metoda UI_runn() objektu typu UI_main, které předáme jako parametr celou instanci hry (objekt typu game_list_atr) a také index hráče, jež je ovládán programově. Na základě těchto parametrů následně UI[1] provede požadované operace.

Kontrola tahů UI[1] se provádí vždy ještě před distribucí dat na hráče. Pro bližší představu můžeme implementaci do stávající struktury znázornit vývojovým diagramem:



Obrázek 11: Zapojení do procesu

[1]UI = umělá inteligence

6 Implementace klienta

6.1 Základní funkční princip

Postupy na práci s daty a komunikací jsou dva:

- servisní - Obsahuje aktivity: přihlášení do hry, vytvoření a připojení do hry, obnova seznamu her.
- game - Vyjadřuje kompletní přístup k otevřené hře a jejím funkčním aspektům.

Dle obecného popisu postupů můžeme vyvodit, že servisní princip zastřešuje všechny akce a metodiky mimo hraní hry a její komunikace s hrou, pro což je využíván princip game.

6.1.1 Připojení a založení hry

Abychom mohli vést aktuálně více rozehraných her, které jsou na sobě nezávislé, je zapotřebí pro každou nově vytvořenou hru vytvořit také nové připojení k serveru, tzn. nová proměnná typu TcpClient. V rámci řešení tohoto aspektu byl vytvořen List typu TcpClient, který sdružuje tyto připojení a v případě vytvoření nebo připojení do existující hry předá jako jeden z parametrů objekt reprezentující připojení k serveru typu NetworkStream.
Díky takto dynamicky měnícímu se listu je možné mít rozehráno několik různých her najednou.

6.1.2 Procesy běžící na pozadí

Základním principem realizace této metodiky je použití vláken. V případě řešené problematiky, jak již bylo zmíněno v předchozích kapitolách, je vlákno, jež nám běží na pozadí, obsluhující komunikaci hrané hry se serverem. Hlavním důvodem využití je možnost separace činností, tzn. hlavní vlákno, které se stará o vytvoření a správu formuláře, je stále k dispozici pro aktivní operace. V případě nevyužití speciálního vlákna běžícího na pozadí, jen pouhou komunikací by došlo k zablokování hlavního procesního vlákna, které neustále testuje, zda server odeslal data, což by mělo za následek nefunkčnost GUI základny (tlačítka, apod.).

V našem případě na pozadí běží metoda wait_to_data(), která deserializuje přijatý objekt a ihned zavolá aktualizaci herní plochy (detail v sekci **Aktualizace herní plochy**). Následně po aktualizaci dojde ke zpracování obdržených dat, tzn. ověří, zda padlo číslo 7, v případě rozšíření Námořníci také probíhá ověření příznaku golden_river (surovina zlatá řeka detail v sekci **Pravidla a jejich rozšíření**). Logika posloupností je téměř totožná s logikou, která byla popsána v sekci Umělá inteligence, samozřejmě s určitými modifikacemi.

Mezi tyto modifikace patří obsluha vyčkávání na reakci hráče. Tato obsluha je zajištěna pomocí dočasného pozastavení metody wait_to_data (běžící ve vláknu) do doby, než hráč provede potřebnou akci. Typickým představitelem je ztržení surovin při hodu čísla 7, kdy je zapotřebí, aby klient odevzdal povinné suroviny v případě nesplnění podmínek. Průběh vlákna je pozastaven, aby nedocházelo dočasně k příjmu nekorektních dat. V rámci zpracování objektu je také v případě počátku tahu spuštěno odpočítávání zbytkového času do ukončení hry v případě neaktivnosti hráče. Tento odpočet je realizován dalším vláknem běžícím na pozadí, které mění text v zobrazené komponentě.

6.2 Grafické zobrazení

Standardně rozlišujeme dva formuláře: game_form a form1, který reprezentuje formulář pro přihlášení a veškeré akce mimo akcí týkajících se konkrétní hry. Formulář game_form zastřešuje

kompletně herní činnosti společně s komunikací. Grafický vzhled game_form a form1 je obsahem Příloha 2, Příloha 3.

6.2.1 Herní plocha

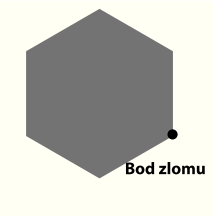
Vykreslování herní plochy je zajištěno dynamicky tvořenými objekty typu PictureBox, jež se navzájem překrývají. Toto překrytí je nutné z důvodu tvaru hracího pole (šestiúhelník). Způsob výpočtu je uveden níže.

Hlavním problémem na vyvíjené platformě je transparentnost pozadí objektů. Představme si dva obrázky. Pozadí formuláři přiřadíme bílou barvu pozadí, pozadí nadřazeného objektu je transparentní a podřazený objekt vyplníme černou barvou. V případě překrytí těchto dvou objektů zjistíme, že ačkoliv jsme podřazený objekt vyplnili černou barvou, stále vidíme v místě překrytí barvu bílou, tzn. barvu hlavního formuláře.

Pro počáteční vykreslení a nastavení umístění jednotlivých objektů slouží metody:

- set_image() - Vygenerování potřebného počtu objektů typu PictureBox a jejich uložení do kolekce. Dle materiálu hracího pole je objektu nastaveno pozadí.
- set_place() - Metoda pro nastavení velikosti a lokace objektu.
- paint_corners() - Vykreslení 2 spodních okrajů v částech, kde se překrývá několik objektů mezi sebou.
- find_pixel() - Nalezení pixelu zlomu (detail uveden níže).
- find_angle() - V případě vyvolání události Click() je hledán úhel zlomu.

Mezi klíčové metody patří převážně metoda find_pixel(), která nám dohledá bod zlomu obrázku nacházejícího se ve zlomu úhlu. Toto dohledání je realizováno postupným porovnáváním barvy pixelu a v případě nalezené neshody je vrácena hodnota pixelu zlomu.

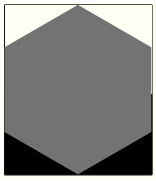


Obrázek 12: Bod zlomu

Pokud známe pixel zlomu, můžeme jednoduše dopočítat, na jakých souřadnicích se mají překrývat mezi sebou objekty reprezentující hrací pole (metoda set_place()).

Pro vyřešení problému definovaného na počátku této sekce slouží metoda paint_corner(). Tato metoda je vyvolána pouze jednou a jejím hlavním účelem je v místech překrytí objektů mezi sebou korektně vykreslit místa překrytí. Jako příklad nám může sloužit černý obdélník, do kterého vložíme bílý šestiúhelník, stejně široký a vysoký. Černá místa jsou plochy překrytí dvou hracích polí. Metoda paint_corner() vykreslí do spodních dvou míst překrytí obrázky (pomocí Graphics a metody DrawImage) hracích polí, která zde mají správně patřit. Výsledek uloží jako pozadí objektu, v němž prováděl změnu, tzn. výsledné pozadí jednoho hracího pole poté vypadá viz.obrázek.

Jelikož postupně takto procházíme všechny objekty reprezentující herní plochu, stačí nám vykreslovat pouze spodní dvě místa překrytí, z důvodu částečného překrytí objektů nižší řady,



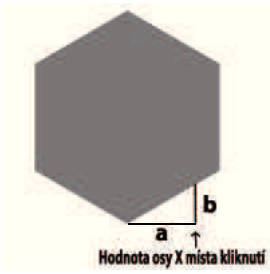
Obrázek 13: Princip vykreslení rohů

objekty řady vyšší. Velmi důležitou vlastností je, aby každá nadřazená řada[1] objektu typu PictureBox měla nastaveno zobrazení v popředí, tzn. bude vždy částečně překrývat řadu[1] nižší.

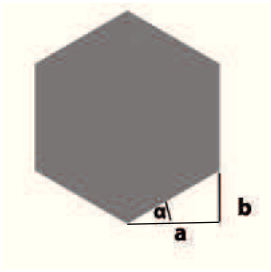
6.2.2 Detekce kliknutí hracího pole

Každé vytvořené hrací pole obsahuje také obsluhu události Click(). Zde probíhá algoritmus pro zjištění části objektu, v němž byl proveden klik myší. V případě kliknutí do 6 úhelníku je obsluhován objekt, který vyvolal metodu Click(). Ovšem klikneme-li do míst překrytí dvou objektů, je zapotřebí vypočítat, kterého objektu se interakce týká.

Postup pro výpočet pixelu je stanoven na základě vypočtení úhlu a strany protilehlé tomuto úhlu.



Obrázek 14: Kliknutí na hrací pole



Obrázek 15: Ukázka počítaného úhlu

Jak vidíme na obrázku, lze aplikovat jednoduchá pravidla pro výpočet stran v pravoúhlém trojúhelníku na základě úhlu. Úhel zjistíme metodou find_angle(), kde aplikujeme výpočet úhlu v pravoúhlém trojúhelníku za předpokladu, že známe obě odvěsny. Pro výpočty jsou tedy použity tyto vzorce:

$$\alpha = Arctan(\frac{b}{a}) \tag{1}$$

a ... šířka objektu typu PictureBox / 2
b ... výška objektu typu PictureBox - X souřadnice pixelu zlomu

$$b = Tan(\alpha) * a \tag{2}$$

a ... X souřadnice pixelu zlomu - šířka objektu typu PictureBox / 2

[1]Řada = několik hracích polí umístěných vedle sebe

Jakmile máme vypočtenu velikost odvěsny b pro konkrétní lokaci kliknutí, můžeme korektně určit, o které hrací pole se jedná (hodnota parametru X nesmí být větší než výška objektu typu PictureBox - velikost odvěsny b).

Počítání úhlů a dohledávání pixelu zlomu je nutné z důvodu automatické změny velikosti objektu PictureBox reprezentujícího jedno hrací pole, v závislosti na celkové velikosti herního okna. Výsledkem takovéto úpravy velikosti je kompletní změna svírajícího úhlu a tudíž nutný přepočet.

6.2.3 Aktualizace herní plochy

Při pohledu můžeme aktualizaci herní plochy rozdělit na dva druhy:

- Změna velikosti formuláře
- Změna na základě dodaných dat

Oba druhy se konkrétně v implementaci liší pouze vyvoláním metody set_place(), která je vyvolána v případě změny velikosti formuláře. Funkce přenastaví parametr Location a Size objektu typu PictureBox na nové hodnoty.

Základní posloupnost pro aktualizaci je jednoznačně dána strukturou metody repaint(), volající funkce objektu typu main_graphics:

- set_city() - Vykreslení vesnic, měst.
- set_street() - Vykreslení cest, lodí.

V případě překreslení jsou stěžejními pomocnými metodami find_pixel() a find_angle() (pro vykreslení cest a lodí). Obecné schéma vykreslení funguje pro výše uvedené metody podobně. Obě metody vykreslují obrázek reprezentující daný typ budovy do okolních objektů typu PictureBox z důvodu vzájemného překrývání těchto objektů mezi sebou. Představme si, že jsme detekovali postavení vesnice na bodu 0, vesnice se tudíž kvůli překrytí objektů vykreslí pouze do 2 objektů nadřazené řady typu PictureBox. Pokud budeme stavět cestu či loď, vykreslení obrázku reprezentující tento typ objektu bude provedeno maximálně do dvou objektu typu PictureBox.

Vykreslování je zajištěno pomocí Graphics a její metody DrawImage. Mezi hlavní nevýhody dynamického vykreslování pomocí DrawImage patří nutné překreslení po provedení posunutí, minimalizaci okna, kdy vykreslené obrázky ihned mizí.

V případě spuštění aktualizace ve vláknu běžícím na pozadí, je při neošetření detekována chyba. Tato chyba vzniká v důsledku úpravy ovládacího prvku vláknem, ve kterém nebyl vytvořen. Ošetření tohoto problému je realizováno pomocí metody Invoke a InvokeRequired. Metoda Invoke přesměrovává volání metody z jiného vlákna do vlákna, v němž byl prvek vytvořen. Druhá zmíněná metoda obstarává verifikaci zda-li je metoda ovládacího prvku volána z vlákna v němž byl prvek vytvořen.

6.2.4 Obchodování

Obchod je rozdělen na dvě přístupová oprávnění:

- Plný přístup
- Omezený přístup

```
private void SetEnabled(Control ctrl, bool status)
{
    if (ctrl.InvokeRequired)
    {
        SetEnabledCallback d = new SetEnabledCallback(SetEnabled);
        this.Invoke(d, new object[] { ctrl, status});
    }
    else
    {
        ctrl.Enabled = status;
    }
}
```

Obrázek 16: Změnu atributu Enabled v ovládacím prvku

Hráč na tahu automaticky disponuje plným režimem oprávnění, tzn. oprávnění pro vytvoření nabídek. Hráč, jenž na tahu není, má možnost pouze tyto nabídky zobrazit, akceptovat a poslat zpět modifikovaný požadavek, vyjadřující jaké suroviny vyžaduje pro akceptování směny (omezený přístup). Na základě těchto přístupových práv se dynamicky generuje komponenta typu Panel, která dle typu oprávnění obsahuje funkční prvky (zobrazení detailu nabídky, možnost úpravy a zrušení nabídky a další). Data pro zobrazení jsou obsažena v objektu typu main_send_data v instanci typu exchange_main, jehož struktura byla popsána v sekci **Implementace serveru**.

7 Závěr

Cílem této práce bylo převedení deskové hry Osadníci z Katanu do podoby aplikace s možností hraní více hráčů a využití hráče s umělou inteligencí. Finální podoba aplikace umožňuje hraní hry ve 4 hráčích, což je maximum dle omezení danými pravidly. Aplikace také umožňuje výběr typu hry dle pravidel. V tuto chvíli je možnost výběru limitována na Základní verzi a rozšíření Námořníci. Implementované řešení se shoduje s pravidly pro konkrétní hru s drobnými obměnami (pořadí hráčů na počátku). Samozřejmostí je databáze klientů a vedení jejich statistik na základě dosažených výsledků.

Aplikace je umístěna na přiloženém CD a je součástí této práce. Pro spuštění obou částí je nutnost mít nainstalován NET Framework 4. Pokud je tato součást systému nainstalována, spuštění serverové i klientské části již proběhne korektně.

Navržená struktura serveru je rozložena tak, aby v případě úpravy jednotlivých částí, tzn. pravidla, řízení umělé inteligence, komunikace, stačilo pouze nahradit stávající třídu upravenou třídou, avšak je zapotřebí stejného konstruktu (názvu metod a návratových typů). V případě změny principu komunikace je zapotřebí adekvátně změnit funkčnost klientské části.

Dalším mým cílem bylo realizovat a zakomponovat do běžících procesů umělou inteligenci. Implementace umělé inteligence byla navržena jako plně nahraditelná, tzn. v případě změny stačí opět pouze zaměnit třídu ovládající umělou inteligenci, přičemž je zde pouze jedna stěžejní metoda, která je volána zvenčí.

8 Literatura

Reference

[1] *B. W. Kernighan, D. M. Ritchie:* **Programovací jazyk C**
Computer Press

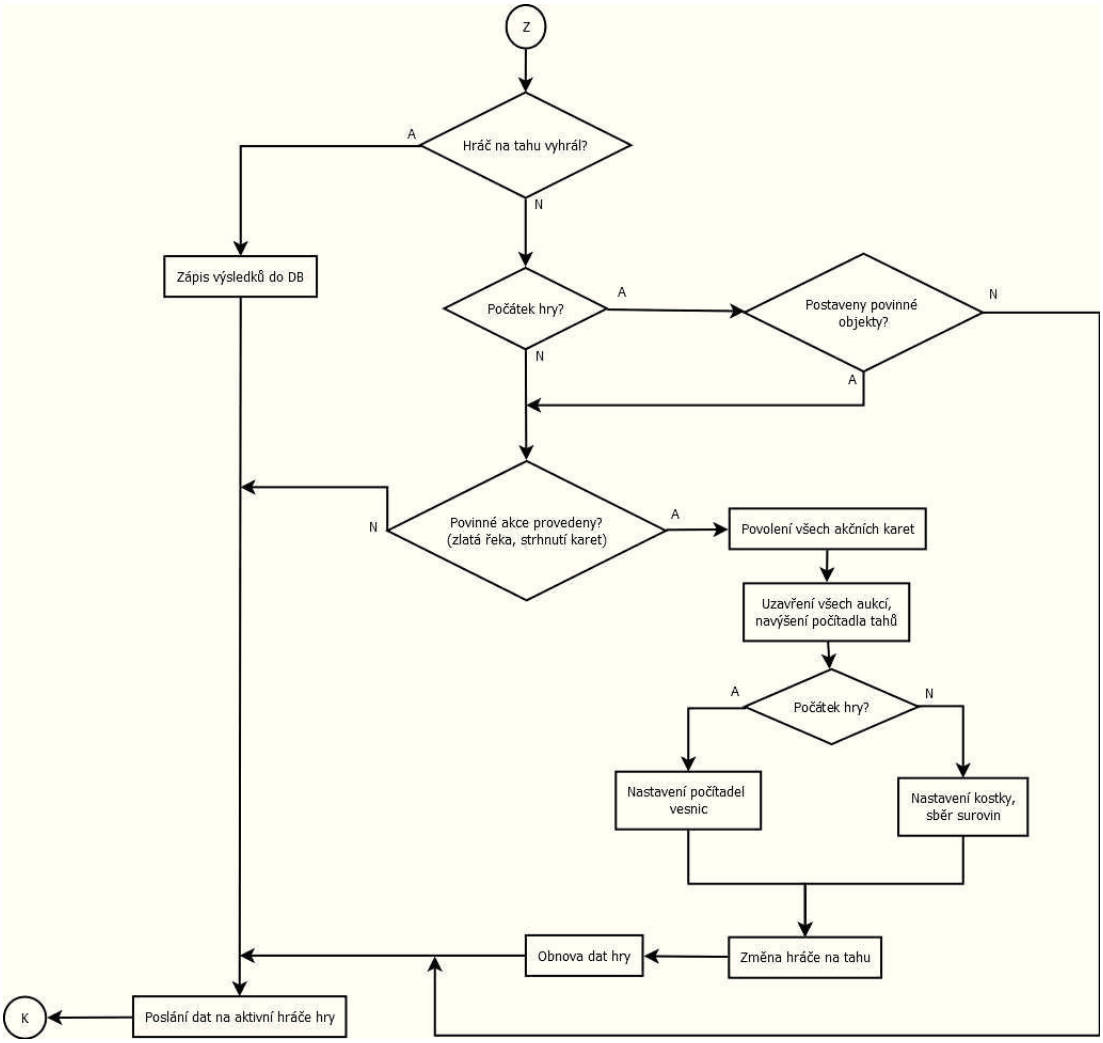
[2] *J. Sharp:* **Microsoft Visual C# 2010**
Computer Press

[3] *Pravidla rozšíření Námořníci*
`http://www.hrajeme.cz/Hrajeme/Files/OsadniciNamornici.pdf`

9 Seznam příloh

9.1 Příloha č.1

Vývojový diagram ukončení tahu



9.2 Příloha č.2

Grafický vzhled game_form

9.3 Příloha č.3

Grafický vzhled form1